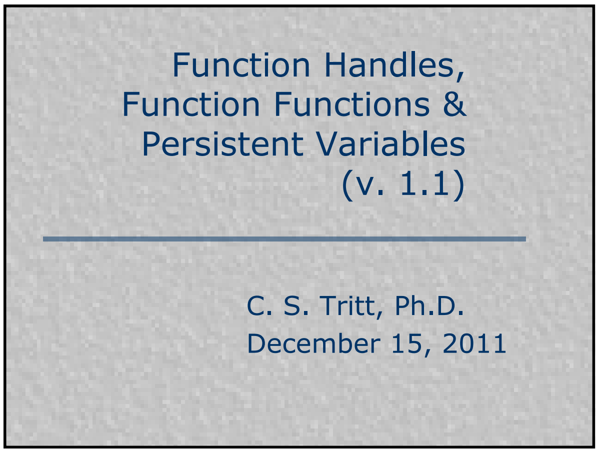
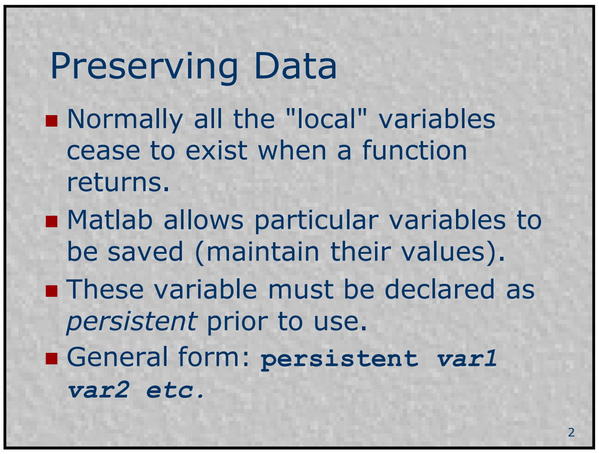


Function Handles, Function Functions & Persistent Variables (v. 1.1)

C. S. Tritt, Ph.D.

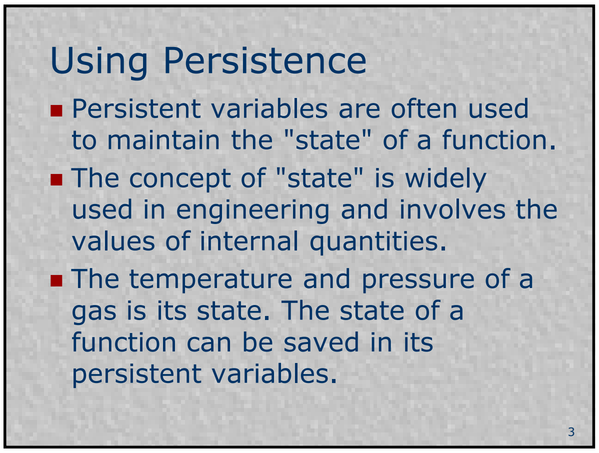
December 15, 2011



Preserving Data

- Normally all the "local" variables cease to exist when a function returns.
- Matlab allows particular variables to be saved (maintain their values).
- These variable must be declared as *persistent* prior to use.
- General form: ***`persistent var1 var2 etc.`***

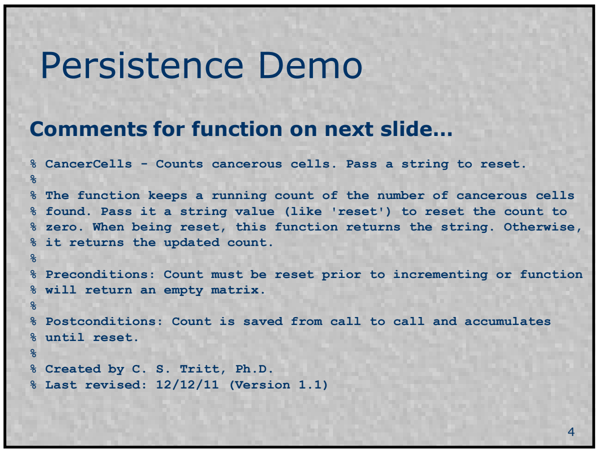
2



Using Persistence

- Persistent variables are often used to maintain the "state" of a function.
- The concept of "state" is widely used in engineering and involves the values of internal quantities.
- The temperature and pressure of a gas is its state. The state of a function can be saved in its persistent variables.

3

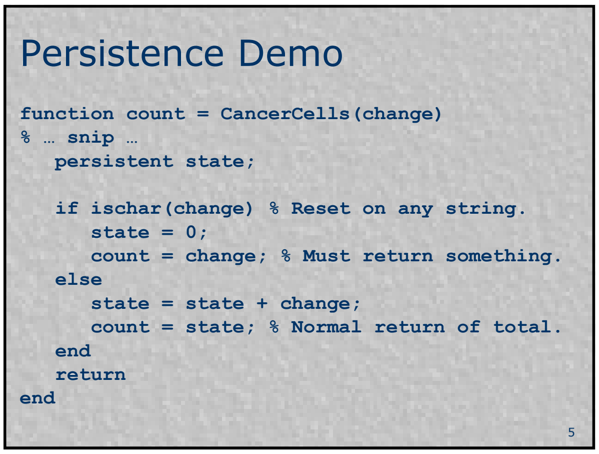


Persistence Demo

Comments for function on next slide…

```
% CancerCells - Counts cancerous cells. Pass a string to reset.
%
% The function keeps a running count of the number of cancerous cells
% found. Pass it a string value (like 'reset') to reset the count to
% zero. When being reset, this function returns the string. Otherwise,
% it returns the updated count.
%
% Preconditions: Count must be reset prior to incrementing or function
% will return an empty matrix.
%
% Postconditions: Count is saved from call to call and accumulates
% until reset.
%
% Created by C. S. Tritt, Ph.D.
% Last revised: 12/12/11 (Version 1.1)
```

4

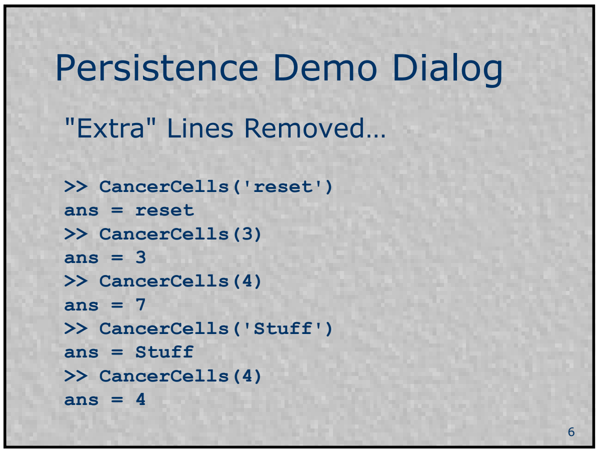


Persistence Demo

```
function count = CancerCells(change)
% … snip …
   persistent state;

   if ischar(change) % Reset on any string.
      state = 0;
      count = change; % Must return something.
   else
      state = state + change;
      count = state; % Normal return of total.
   end
   return
end
```

5



Persistence Demo Dialog

"Extra" Lines Removed…

```
>> CancerCells('reset')
ans = reset
>> CancerCells(3)
ans = 3
>> CancerCells(4)
ans = 7
>> CancerCells('Stuff')
ans = Stuff
>> CancerCells(4)
ans = 4
```

6

Function Handles

- Allow the information needed to execute a particular function to be stored in a variable.
- Created using the `@` operator or ***`str2func`*** function.
- Called directly or using ***`feval`***.
- Used to pass functions as arguments to other functions and in other ways (including for creating GUI's).

7

Function Handle Example

Given the function:

```
function result = sample_func(x)
% Function sample_func implements a simple polynomial.
%
% Created by Dr. C. S. Tritt, 12/11/06

    result = x.^2 - 2.*x + 1;
end % function sample_func.
```

8

Results Produced

```
>> sample_func(3)
ans =
     4

>> hMyHandle = @sample_func
hMyHandle =
    @sample_func
>> hMyHandle(3)
ans =
     4

>> hYourHandle = str2func('sample_func')
hYourHandle =
    @sample_func
>> hYourHandle(3)
ans =
     4

>> feval(hMyHandle, 3)
ans =
     4
```

9

Function Functions

- Functions with input arguments that are **names** of other functions.
- Example
  - fzero – function locates the a zero of a function that is passed to it
  - fzero('cos',[0 pi]) – finds the zero of the cosine function between 0 and pi (which is at pi/2) = 1.57. Note use of quotes.

10

Using *eval* and *feval*

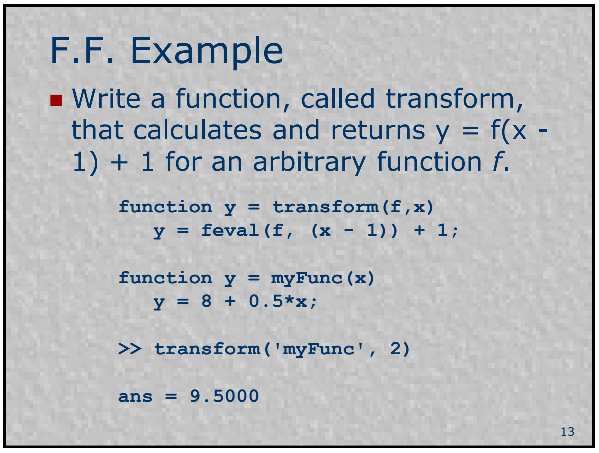
- The key to its operation are two matlab functions eval and feval.
- *eval* evaluates a character string as though it has been typed into the command window:
  - X = eval('sin(pi/4)')
- *feval* evaluates a named function at for the specific input:
  - X = feval('sin', pi/4)

11

Some Matlab F.F.'s

Function Name	**Description**
fminbnd	**Minimize a function.**
fzero	**Finds the zero of a function.**
quad	**Numerically integrate a function.**
ezplot	**Easy to use function plotting**
fplot	**Plot a function by name.**

12



F.F. Example

- Write a function, called transform, that calculates and returns y = f(x - 1) + 1 for an arbitrary function *f*.

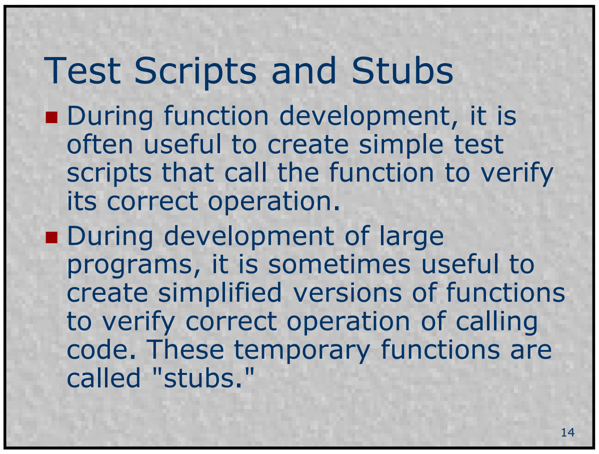
```
function y = transform(f,x)
   y = feval(f, (x - 1)) + 1;

function y = myFunc(x)
   y = 8 + 0.5*x;

>> transform('myFunc', 2)

ans = 9.5000
```

13



Test Scripts and Stubs

- During function development, it is often useful to create simple test scripts that call the function to verify its correct operation.
- During development of large programs, it is sometimes useful to create simplified versions of functions to verify correct operation of calling code. These temporary functions are called "stubs."

14